TRANSMITTAL OF APPEAL BRIEF (Large Entity)	Docket No. ITL-0278-US

In Re Application Of:

Steven C. Dake; Paul E. Luse

Serial No.	Filing Date	Examiner	Group Art Unit
09/469,277	December 22, 1999	Dwin M. Craig	2123

Invention:

Method for Modeling Hardware Using Software

RECEIVED
MAY 01 2003
Technology Center 2100

TO THE ASSISTANT COMMISSIONER FOR PATENTS:

Transmitted herewith in triplicate is the Appeal Brief in this application, with respect to the Notice of Appeal filed on

The fee for filing this Appeal Brief is: $320.00

☒ A check in the amount of the fee is enclosed.

☐ The Commissioner has already been authorized to charge fees in this application to a Deposit Account. A duplicate copy of this sheet is enclosed.

☒ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. 20-1504
A duplicate copy of this sheet is enclosed.

RECEIVED
2003 APR 30 AM 10:00
BOARD OF PATENT APPEALS AND INTERFERENCES

[illegible]
Signature

Dated: April 24, 2003

Mark J. Rozman, Registration No. 42,117
Trop, Pruner & Hu, P.C.
8554 Katy Freeway, Suite 100
Houston, Texas 77024

21906
PATENT TRADEMARK OFFICE

I certify that this document and fee is being deposited on 04/24/2003 with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

[illegible]
Signature of Person Mailing Correspondence

Jennifer Juarez
Typed or Printed Name of Person Mailing Correspondence

cc:

P30LARGE/REV03

#10/Appl Brief 6-12-03 [illegible]

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	STEVEN C. DAKE; PAUL E. LUSE	§	Group Art Unit:	2782
Serial No.:	09/469,277	§		
		§	Examiner:	Dwin M. Craig
Filed:	December 22, 1999	§		
For:	METHOD FOR MODELING HARDWARE USING SOFTWARE	§	Atty. Dkt. No.:	INTL-0278-US (P7627)

RECEIVED
2003 APR 30 AM 10:01
BOARD OF PATENT APPEALS AND INTERFERENCES

Board of Patent Appeals & Interferences
Commissioner for Patents
Washington, D.C. 20231

APPEAL BRIEF

Sir:

Applicants respectfully appeal from the final rejection mailed December 2, 2002.

I. REAL PARTY IN INTEREST

The real party in interest is the assignee Intel Corporation, the assignee of the patent application by virtue of the assignment recorded at Reel/Frame 010509/0494.

II. RELATED APPEALS AND INTERFERENCES

None.

III. STATUS OF THE CLAIMS

The application was originally filed with claims 1-27. Claims 28-32 were added by amendment. Claims 1-21 and 28-32 remain pending. Claims 1-2, 6-10, 16-21, and 28-32 are the subject of this appeal. The rejection of claims 3-5 and 11-15 is not being appealed.

IV. STATUS OF AMENDMENTS

On January 17, 2003, Applicants filed a Reply to the Final Rejection in which claims 1 and 29 were amended and claims 2-5, 11-15 and 28 were cancelled without prejudice. In an

05/01/2003 [illegible] 00000031 09469277
01 FC:1402 320.00 OP

Date of Deposit: April 24, 2003
I hereby certify under 37 CFR 1.8(a) that this correspondence is being deposited with the United States Postal Service as **first class mail** with sufficient postage on the date indicated above and is addressed to the Board of Patent Appeals & Interferences, Commissioner for Patents, Washington, DC 20231.
[illegible]
Jennifer Juarez

Advisory Action mailed February 19, 2003, the Examiner refused to enter the proposed amendments. Accordingly, claims 1-21 and 28-32 remain pending.

V. SUMMARY OF THE INVENTION

In general, in one embodiment of the invention, a method comprises defining a plurality of hardware devices as a plurality of objects, providing a plurality of tools to perform a plurality of operations on the plurality of objects, executing a software program to use the tools and responding to the plurality of operations by the plurality of hardware devices.

Turning to Figure 1, the object model application programming interface (API) 10, according to one embodiment of the invention, may include a configuration library 20 and a plurality of objects 30. The object model API 10 is coupled between a software program 12 and a hardware device 16, a hardware device 17, and a hardware device 18.

The hardware device 16 may, for example, be supplied by a different vendor than the hardware device 17, etc. The hardware device 16 may itself include software, such as a software interface (not shown). The hardware device 16 may operate using one protocol, the hardware device 17 may operate using a second protocol, and so on. See Specification, pp. 3-4.

The object model API 10 may reside between the software program 12 and the hardware devices 16-18. In one embodiment of the invention, the object model API 10 provides the capability to abstract the hardware devices 16-18 so that the software program 12 need not be written with the distinct requirements of the hardware devices 16-18 in mind. Instead, the software program 12 communicates with the objects 30 which model the hardware devices 16-18. In turn, the hardware devices 16-18 communicate with the object model API 10, whether to take actions, provide information, or perform other operations on the hardware devices 16-18.

The object model API may be beneficial for some hardware/software models. The software program 12 may be written with less than full knowledge of the particular hardware involved. Instead, using the object model API 10, the software program 12 may be written once a "model" of the hardware devices 16-18 has been defined. The objects 30 represent this model.

Each hardware device 16-18 to be supported by the software program 12 communicates with the object model API 10. Where the hardware devices 16-18 include a software interface, the interface may communicate with the object model API 10. For example, the interface to the hardware devices 16-18 may respond to commands from the object model API 10 such that the commands are executed upon the hardware devices 16-18.

As part of the object model API 10, the configuration library 20 includes a set of tools which enable the software program 12 to communicate with the objects 30. The objects 30 model the hardware devices 16-18. Once the software program 12 knows how to retrieve and manipulate the objects 30, using the tools from the configuration library 20, the software program 12 may perform operations on the objects 30. See Specification, pp. 4-5.

The objects 30, in essence, define the hardware devices 16-18 by modeling the operations (or methods), the attributes (or properties), and the actions (or events) of each hardware device 16-18. An operation, known as a method, may be invoked upon an object. The method is appropriate for the real-world hardware modeled by the object. One or more attributes, described below as properties, may also be used to define an object. Actions, described below as events, may model real-world occurrences by the hardware devices 16-18.

In Figure 2, in one embodiment of the invention, the object 30 includes methods 32, properties 34 and events 36 for representing the characteristics of the hardware devices 16-18 modeled by the object 30. The methods 32 may include functions or actions which the hardware devices 16-18 modeled by the object 30 may perform.

In one embodiment of the invention, the configuration library 20 may be used to communicate with the objects 30. In Figure 3, the tools of the configuration library 20 may include several functions. The functions may be used by the software program 12 to communicate with the objects 30.

The configuration library 20 includes an initialization function 52. The initialization function 52 initializes the configuration library 20. In one embodiment of the invention, the initialization function 52 is executed prior to any other function in the configuration library 20.

A second initialization function 54 allows the software program 12 to override any memory management functions of the environment. Some system management architectures provide unique memory management routines. Under such system management architectures, memory management routines may be replaced with other routines. The second initialization function 54 thus overrides the memory manager upon command.

The configuration library 20 further includes a set of functions for object creation and object discovery. An object locating function 58 provides the capability to locate an object of a specific type with a specific property value using a comparison operator as an input value. The object locating function 58 may be used to determine the components of the hardware device 16 by the software program 12, for example. Closely associated with the object locating function 58, a subsequent object locating function 60 provides the capability to locate the next object based upon the parameters provided in the object locating function 58. See Specification, pp. 5-8.

The object model API 10 may be used as a front end to a variety of types of hardware devices 16-18. By distilling the hardware devices 16-18 into distinct components, an object model 30 for each component may then be created. A variety of hardware devices may then be defined in terms of operations (methods), attributes (properties) and actions (events). These methods 32, properties 34, and events 36 are then encapsulated as objects 30. To support the hardware devices 16-18, a developer may then use the functions of the configuration library 20, described above, to perform operations upon the objects 30.

Because the software program 12 does not communicate directly with the hardware devices 16-18, the requirements of the hardware devices 16-18 are unimportant to the software program 12. Instead, the software program 12 communicates with the objects 30 using the tools of the configuration library 20. In this way, the software program 12 may support the hardware devices 16-18, even when the hardware devices 16-18 change. Further, the software program 12 may support future hardware additions.

Software which supports a redundant array of independent disks, or RAID, system, may benefit from the object model API 10. A RAID system is essentially a collection of disks, connected by one or more buses, and employing one or more controllers. RAID systems may employ one or more disk drives, typically to provide fault tolerance and to enhance performance. In addition to the disk drives themselves, the RAID system may include a plurality of controllers and buses. Further, the RAID system may be organized into physical representations of disks, called arrays, or logical representations of disks, called volumes. The RAID hardware, as well as its supporting software, may therefore be complex. See Specification, pp. 9-10.

Turning to Figure 4, a particular implementation of the object model API 10, shown as a RAID object API 10a, includes the configuration library 20 and RAID objects 30a, in accordance with one embodiment of the invention. The RAID API 10a interfaces with a RAID software program 12a and a RAID hardware device 16a, a RAID hardware device 17a, and a RAID hardware device 18a.

As in Figure 1, the RAID hardware devices 16a-18a may include RAID hardware device 16a, from one vendor, RAID hardware 17a, possibly from a second vendor, and RAID hardware 18a, possibly from a third vendor. Each RAID hardware device 16a-18a may include a software interface (not shown). The RAID hardware devices 16a-18a are responsive to directives from the RAID API 10a. Likewise, the RAID software 12a communicates with the objects 30a modeling the RAID hardware devices 16a-18a using the configuration library 20.

In Figure 5, the RAID hardware devices 17a include a plurality of arrays 130. For example, disks 120C and disks 120D are joined to form array 130B. In one embodiment of the invention, each array 130 has at least one volume 140 associated with the array 130. In Figure 5, the array 130D includes two volumes 140E and 140F. However, array 130B includes only a single volume 140D.

From the physical RAID hardware devices 17a of Figure 5, a plurality of RAID objects 30a may be derived. In Figure 6, the RAID objects 30a include controller objects 100, bus objects 110, disk objects 120, array objects 130, and volume objects 140.

Each of the RAID objects 30a of Figure 6 represents a component of the RAID hardware devices 17a in Figure 5. For example, the RAID hardware devices 17a include eight disks 120A-120H. Thus, the RAID objects 30a include eight disk objects 120a-120h. Likewise, the RAID hardware devices 17a include two controllers 100A and 100B. Accordingly, in Figure 6, controller object 100a and 100b are two of the RAID objects 30a. The RAID objects 30a thus represent, using objects, the entire configuration of the RAID hardware devices 17a. See Specification, pp. 10-12.

In Figure 8, the bus object 110 includes a scan bus method 111 and several properties 34, including a bus indexing property 112, a bus identification property 113, a bus protocol property 114, a bus device count property 115 and a Boolean property 116, which tells whether the bus is presently scanning. As with the controller object 100, the bus object 110 includes no events for the RAID software 12a to monitor. However, the RAID software 12a may set or retrieve any of the properties or may invoke the scan bus method 111, as desired.

In Figure 9, the disk object 120 includes methods 32, properties 34, and events 36 which allow the RAID software 12a to communicate with the plurality of disks, such as the eight disks 120A-120H of Figure 5. For example, a method 122, a method 125, and a method 126 may be used to mark a disk as normal, offline, and online, respectively. A total block count property 153 may be used to retrieve the total block count of the disk.

In addition to several methods 32 and properties 34, the disk object 120 includes events 36 which may permit the RAID software 12a to monitor the behavior of each disk modeled by the disk object 120. For example, a disk is normal event 161 may permit the RAID software 12a to be notified when the disk has been marked as normal (following the invocation of the mark as normal method 122).

In Figures 10 and 11, the array object 130 and the volume object 140, like the disk object 120, include methods 32, properties 34 and events 36. In one embodiment of the invention, the array is defined as a physical grouping of one or more disks. Accordingly, the array object 130 includes a method 131 to create an array and a method 132 to expand an array. A disk count property 134 enables the RAID software 12a to identify the number of disks 120 included in the array 130. A volume degraded event 180 permits the RAID software 12a to receive notification when a volume 140 of the array 130 has degraded.

The volume object 140 (Figure 11) includes methods such as a method 141 and a method 142, for creating and deleting a volume 140, respectively, from the RAID hardware devices 17a. Properties such as number of bytes per block 171 and total number of blocks 172 provide other indicia about the volume 140. The volume object 140 further includes event notification for failed, migrating, and initializing volumes, an event 181, an event 182, and an event 183, respectively.

Using the above objects in conjunction with the functions of the configuration library 20, the RAID software 12a may perform a number of operations on the RAID hardware devices 17a. Simply by controlling the properties 34 and methods 32 of the RAID objects 30a, the RAID software 12a can configure the RAID hardware devices 17a. See Specification, pp. 12-15.

Thus, an interface to hardware includes a configuration library and objects to model the hardware. Software programs using the interface need not understand how to communicate with the hardware. Instead, the software programs may communicate with the interface. In turn, the interface communicates with the hardware. Such an organization allows the software to be written even before the hardware is implemented, prior to the addition of new or different hardware, or under other conditions in which the hardware requirements are unknown.

VI. ISSUES

A. Are Claims 1-2 Patentable Under 35 U.S.C. § 102(e) Over Savitzky?

B. Are Claims 6-10 Patentable Under 35 U.S.C. § 103(a) Over Brumley In View of Muller?

C. Are Claims 16-18 Patentable Under 35 U.S.C. § 103(a) Over Brumley In View of Morris?

D. Are Claims 19-21 Patentable Under 35 U.S.C. § 103(a) Over Brumley In View of Morris?

E. Is Claim 28 Patentable Under 35 U.S.C. § 103(a) Over Savitzky In View of Muller?

F. Are Claims 29 and 31-32 Patentable Under 35 U.S.C. § 103(a) Over Savitzky In View of Muller?

G. Is Claim 30 Patentable Under 35 U.S.C. § 103(a) Over Savitzky In View of Muller and Further View of Christianson?

H. Are Claims 1 and 2 Enabled Under 35 U.S.C. § 112?

I. Are Claims 6-10 Enabled Under 35 U.S.C. § 112?

J. Is Claim 16 Enabled Under 35 U.S.C. § 112?

K. Are Claims 17 and 19-21 Enabled Under 35 U.S.C. § 112?

VII. GROUPING OF THE CLAIMS

For purposes of this appeal, Applicants have grouped together claims 1-2; claims 6-10; claims 16-18; claims 19-21; and claims 29 and 31-32, as set forth above for the prior art rejections; and claims 1-2; claims 6-10 and claims 19-21 may be grouped for the enablement rejections.

VIII. ARGUMENT

A. Claims 1-2 Are Patentable Under 35 U.S.C. § 102(e) Over Savitzky

Claim 1 recites a method including defining a plurality of hardware devices as a plurality of objects; providing a plurality of tools to perform a plurality of operations on the plurality of objects; executing a software program to use the plurality of tools; and responding to the plurality of operations by the plurality of hardware devices.

Claims 1 and 2 stand rejected under § 102(e) over U.S. Patent No. 5,732,261 (Savitzky). This rejection is improper as Savitzky does not disclose "defining a plurality of hardware devices as a plurality of objects." In this regard, the Examiner cites col. 3 lns. 60-67 of Savitzky. Final Office Action, p. 3. However Savitzky does not disclose this element of claim 1, as instead Savitzky relates to a "software object that represents the services and state of some remote machine." Savitzky, col. 3 lns. 60-62 (emphasis added). Representing such services or state does not define hardware devices as objects. Thus Savitzky does not disclose, at least, "defining a plurality of hardware devices as a plurality of objects." Accordingly, claims 1 and 2 are patentable over Savitzky and the rejection should be reversed.

B. Claims 6-10 Are Patentable Under 35 U.S.C. § 103(a) Over Brumley In View of Muller

Claim 6 recites an article that causes a system to receive a request from a software program; act upon a plurality of objects based upon the request received; and manipulate a redundant array of independent disks (RAID) modeled by the plurality of objects.

Claims 6-10 stand rejected under 35 U.S.C. § 103(a) over U.S. Patent No. 5,926,775 (Brumley) in view of U.S. Patent No. 6,247,077 (Muller). This rejection is improper. The Examiner concedes that nowhere does Brumley disclose a medium storing instructions to "manipulate a redundant array of independent disks modeled by the plurality of objects" as recited by claim 6. Final Office Action, p. 29.

However, the Examiner contends that Muller discloses such a redundant array of independent disks modeled by a plurality of objects. Id. Applicants respectfully disagree. In this regard, while Muller discloses an array of independent disks, nowhere does Muller teach or suggest that such an array be modeled by a plurality of objects, nor manipulation of such modeled objects. Nor is there any motivation to combine Brumley with Muller. Thus claim 6 and claims 7-10 depending therefrom are patentable over the proposed combination.

C. Claims 16-18 Are Patentable Under 35 U.S.C. § 103(a) Over Brumley In View of Morris

Claim 16 recites a system that includes a memory having software which models a plurality of disks as disk objects; provides a plurality of tools for performing operations on the disk objects; and invokes a response by the plurality of disks to the operations performed on the disk objects.

Claims 16-18 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Brumley in view of U.S. Patent No. 5,877,966 (Morris). This rejection is improper at least because there is no suggestion or motivation to combine the references. Brumley relates to software architecture for a data acquisition system, whereas Morris relates to a configurator and use thereof to generate configurations, such as configurations of end products or services. See Morris, col. 1, lns. 10-15. As such there is no motivation to combine these references from disparate fields.

More so, the portions of Morris cited, namely figure 2 and col. 5, lns. 29-39 and 54-57 (see Final Office Action, pp. 31-32), merely relate to creating a configuration of a personal computer having a hard disk drive contained therein. Nowhere does Morris teach or suggest software which "models the plurality of disks as a plurality of disk objects" nor "provides a plurality of tools for performing a plurality of operations on a plurality of disk objects" nor "invokes a response by the plurality of disks to the plurality of operations performed on the plurality of disk objects".

Further, as conceded by the Office Action, nor does Brumley teach or suggest this software, as Brumley relates to a data acquisition device, not a plurality of disks. Final Office Action, p. 31. For at least these reasons, claim 16 and claims 17 and 18 depending therefrom patentably distinguish over the proposed combination.

D. Claim 19-21 Are Patentable Under 35 U.S.C. § 103(a) Over Brumley In View of Morris

Claim 19 depends from claim 18 which in turn depends from claim 16 discussed above (See Section VIII.C). Claim 19 further recites that the memory stores software which models a plurality of buses as a plurality of bus objects; and models a plurality of controllers as a plurality of controller objects.

Claim 19 stands rejected under § 103(a) over Brumley in view of Morris. Applicants respectfully disagree that Brumley teaches or suggests multiple bus objects and multiple controller objects. In this regard, the figures cited by the Examiner, namely FIGS. 11-13, nowhere teach or suggest modeling a plurality of buses as a plurality of bus objects nor modeling a plurality of controllers as a plurality of controller objects, as recited by claim 19. Thus, claim 19 and claims 20 and 21 depending therefrom are patentable over the proposed combination and the rejection should be reversed.

E. Claim 28 Is Patentable Under 35 U.S.C. § 103(a) Over Savitzky In View of Muller

Claim 28 depends from claim 1 and further recites that the plurality of hardware devices comprises a redundant array of independent disks.

Claim 28 stands rejected under §103(a) over Savitzky in view of Muller. The Examiner concedes that Savitzky does not teach or suggest a redundant array of independent disks. Final Office Action, p. 32. More so as discussed above (See Section VIII.B), Muller does not teach or suggest defining redundant array of independent disks as a plurality of objects. Thus claim 28 is patentable over the combination and the rejection should be reversed.

F. Claims 29 and 31-32 Are Patentable Under 35 U.S.C. § 103(a) Over Savitzky In View of Muller

Claim 29 depends from claim 28 and further recites that the plurality of objects comprise a redundant array of independent disk objects. Savitzky does not teach or suggest a redundant array of independent disks, as conceded by the Examiner. Final Office Action, p.32. Further, nowhere does Muller teach or suggest a "plurality of objects [which] comprise redundant array of independent disk objects." Thus there is no teaching or suggestion to combine Savitzky with Muller to obtain the method of claim 29, and claims 29 and 31-32 depending therefrom are patentable and the rejection should be reversed.

G. Claim 30 Is Patentable Under 35 U.S.C. § 103(a) Over Savitzky In View of Muller and Further View of Christianson

Claim 30 depends from claim 29 and further recites that the redundant array of independent disk objects comprises at least one disk object, at least one array object and at least one volume object.

Claim 30 stands rejected under 35 U.S.C. § 103(a) over Savitzky in view of Muller and in further view of U.S. Patent No. 5,915,253 (Christianson). The proposed combination of Savitzky and Muller and Christiansen does not teach or suggest RAID objects comprising "at least one disk object, at least one array object and at least one volume object", as recited by claim 30. As discussed above (See VIII.E), Savitzky and Muller do not teach or suggest RAID objects, and certainly do not teach or suggest such RAID objects comprising at least one disk, array and volume object.

Further, Christiansen does not teach or suggest any multiple disk arrangement and certainly not a redundant array of independent disks. Christiansen also does not teach or suggest array objects whatsoever, or at least one volume object of "a redundant array of independent disk objects." Thus claim 30 is patentable over the proposed combination and the rejection should be reversed.

H. Claims 1 and 2 Are Enabled Under 35 U.S.C. § 112

Claims 1 and 2 stand rejected under 35 U.S.C. § 112 as lacking enablement. In rendering the rejection, the Examiner grossly misperceives the standard of enablement and as such the rejection cannot stand. In the Final Office Action, the Examiner appears to assert that that the alleged lack of enablement relates to unclaimed features. Final Office Action, p. 9-10. For example, the Examiner asserts that a "wrapper," a Unified Modeling Language diagram, source code, and function prototypes, are missing. Id. However, these features and are not claimed and are thus irrelevant for purposes of enablement under 35 U.S.C. § 112.

More so, the Examiner admits that certain of the alleged missing unclaimed features, such as a "wrapper" are "well known in the art." Final Office Action, p. 9. As such, these features need not be present, as a specification need not disclose and preferably omits that which is well-known to those skilled in the art. *In re Buchner*, 18 USPQ2d 1331, 1332 (Fed. Cir. 1991).

With respect to the Examiner's statement that there are no source code examples, it is respectfully submitted that such examples are unnecessary, as source code need not be present in order to enable a specification. *See Fonar Corp. v. General Electric*, 41 USPQ2d 1364 (Fed. Cir. 1997). This is particularly so, as a flow chart and numerous block diagrams are provided.

More so, the Examiner "does not provide any factors, reasons, or evidence" that the specification fails to comply with §112. Instead, the Examiner merely cites portions of the specification that specifically support and enable both claims 1 and 2, and conclusorily states that the "specification does not support in sufficient and clear detail...and is therefore not enabled." *E.g.*, Final Office Action, pp. 11-13 (citing Specification, pp. 3-9 and 12-16 and associated Figures). As such, the rejection is improper and this rejection of claims 1 and 2 should be reversed.

I. Claims 6-10 Are Enabled Under 35 U.S.C. § 112

Claims 6-10 stand rejected under 35 U.S.C. § 112 as lacking enablement. As discussed above (see VIII.H), the Examiner grossly misperceives the standard of enablement and as such the rejection cannot stand. The Examiner "does not provide any factors, reasons, or evidence" that the specification fails to comply with §112.

For example, the Examiner claims that the specification does not define how "a processor based system receives a request from [a] software program and therefore is not enabled." Final Office Action, p. 15. Such sending of requests from a software program to a processor-based system is clearly well-known and need not be described in detail. Further, the Examiner provides no factors as to why such a well-known process is not enabled. Instead, the Examiner merely cites portions of the specification that specifically support and enable claims 6-10, and conclusorily states that the "specification does not support in sufficient and clear detail...and is therefore not enabled." *E.g.*, Final Office Action, pp. 15-16. As such, the rejection is improper and this rejection of claims 6-10 should be reversed.

J. Claim 16 Is Enabled Under 35 U.S.C. § 112

Claim 16 stands rejected under 35 U.S.C. § 112 as lacking enablement. As discussed above (see VIII.H), the Examiner grossly misperceives the standard of enablement and as such the rejection cannot stand. The Examiner "does not provide any factors, reasons, or evidence" that the specification fails to comply with §112. Instead, the Examiner merely cites portions of the specification that specifically supports and enables claim 16 and conclusorily states that the "specification does not support in sufficient and clear detail...and is therefore not enabled." *E.g.*, Final Office Action, p. 19. As such, the rejection is improper and this rejection should be reversed.

K. Claims 17 and 19-21 Are Enabled Under 35 U.S.C. § 112

Claims 17 and 19-21 stand rejected under 35 U.S.C. § 112 as lacking enablement. As discussed above (see VIII.H), the Examiner grossly misperceives the standard of enablement and as such the rejection cannot stand. The Examiner "does not provide any factors, reasons, or evidence" that the specification fails to comply with §112.

For example, with regard to claim 17 the Examiner states that the specification "does not support in clear detail how the software program is stored in memory and is therefore not enabled." Final Office Action, p. 20. Of course, it is well-known how a software program is stored in memory; furthermore the specification includes a block diagram of a system showing software stored in memory (see, e.g., FIG. 5). The Examiner used the same rejection for claims 19, 20 and 21.

As such storage of software programs is well-known, it need not be described further, particularly where the Examiner does not provide any "factors, reasons, or evidence" why the specification fails to comply with § 112. Instead, the Examiner merely cites portions of the specification that specifically support and enable claims 17 and 19-21, and conclusorily states that the "specification does not support in sufficient and clear detail...and is therefore not enabled." *E.g.*, Final Office Action, pp. 20-21. As such, the rejection is improper and this rejection of claims 17 and 19-21 should be reversed.

IX. CONCLUSION

Since the rejections of the claims are baseless, they should be reversed.

Respectfully submitted,

[illegible]

Date: April 24, 2003

Mark J. Rozman, Reg. No. 42,117
TROP, PRUNER & HU, P.C.
8554 Katy Fwy, Ste 100
Houston, TX 77024-1805
512/418-9944 [Phone]
713/468-8883 [Facsimile]

21906
PATENT TRADEMARK OFFICE

APPENDIX OF CLAIMS

The claims on appeal are:

1. A method, comprising:

defining a plurality of hardware devices as a plurality of objects;

providing a plurality of tools to perform a plurality of operations on the plurality of objects;

executing a software program to use the plurality of tools; and

responding to the plurality of operations by the plurality of hardware devices.

2. The method of claim 1, wherein defining the plurality of hardware devices as a plurality of objects further comprises:

assigning a plurality of properties to the plurality of hardware devices; and

assigning a plurality of methods to the plurality of hardware devices.

6. An article comprising a medium storing instructions that cause a processor-based system to:

receive a request from a software program;

act upon a plurality of objects based upon the request received; and

manipulate a redundant array of independent disks modeled by the plurality of objects.

7. The article of claim 6, further storing instructions that cause a processor-based system to use a plurality of configuration library tools to act upon a plurality of objects.

8. The article of claim 7, further storing instructions that cause a processor-based system to invoke a plurality of methods of the plurality of objects.

9. The article of claim 7, further storing instructions that cause a processor-based system to retrieve a plurality of properties of the plurality of objects.

10. The article of claim 7, further storing instructions that cause a processor-based system to monitor a plurality of events for the plurality of objects.

16. A system, comprising:

a processor;

a plurality of disks; and

a memory storing software which:

models the plurality of disks as a plurality of disk objects;

provides a plurality of tools for performing a plurality of operations on the plurality of disk objects; and

invokes a response by the plurality of disks to the plurality of operations performed on the plurality of disk objects.

17. The system of claim 16, wherein the software program is stored in the memory.

18. The system of claim 16, further comprising:

a plurality of buses; and

a plurality of controllers.

19. The system of claim 18, further comprising a memory storing software which:

models the plurality of buses as a plurality of bus objects; and

models the plurality of controllers as a plurality of controller objects.

20. The system of claim 19, further comprising a memory storing software which:

models the plurality of volumes as a plurality of volume objects; and

models the plurality of arrays as a plurality of array objects.

21. The system of claim 20, further comprising a memory storing software which invokes a response to the plurality of operations by:

the plurality of buses for operations performed on the plurality of bus objects; and

the plurality of controllers for operations performed on the plurality of controller objects.

28. The method of claim 1, wherein the plurality of hardware devices comprise a redundant array of independent disks.

29. The method of claim 28, wherein the plurality of objects comprise redundant array of independent disk objects.

30. The method of claim 29, wherein the redundant array of independent disk objects comprises at least one disk object, at least one array object and at least one volume object.

31. The method of claim 29, wherein each one of the redundant array of independent disk objects comprises at least one method, at least one property, and at least one event.

32. The method of claim 31, further comprising communicating with at least one of the redundant array of independent disks using the at least one property.